Chapter 2

Nouvelles Techniques d'Extraction de Chemins à l'Aide du *Fast-Marching*

Résumé — Ce chapitre contient diverses améliorations de la méthode originale du chapitre 1, valables aussi bien en 2D qu'en 3D.
Nous commençons par présenter en section 2.1 une extension au 3D de la méthode classique. Nous détaillons en section 2.2 des méthodes pour accélerer l'extraction de chemins et la rendre plus facile, en réduisant les interactions nécessaires. Nous developpons une méthode pour extraire des chemins centrés dans des structures tubulaires en section 2.3. En section 2.4 nous calculons des trajectoires pour des objets en mouvement, en introduisant l'angle comme dimension supplémentaire au problème. Finalement, nous expliquons l'introduction d'un facteur de récursivité dans le *Fast-Marching* , afin d'extraire des chemins plus longs. Chaque technique est illustrée par un exemple sur une image réelle ou de synthèse.

Abstract — This chapter will detail various improvements and modification of the original method of chapter 1 that are useful for image analysis in 2D as well as in 3D.
In section 2.1, we extend the method detailed in section 1.3 for 2D images to 3D. In section 2.2 we give details about our techniques to make the path extraction scheme faster and easier, by reducing the user interaction. In section 2.3 we develop a new method to extract a path centered in a tubular structure. In section 2.4 we compute trajectories for moving objects, introducing a degree of freedom on their angulation. Finally, in section 2.5, we explain the introduction of a recursivity factor in the *Fast-Marching* algorithm, which enables to extract longer paths. Synthetic and real medical images are used to illustrate each contribution.

2.1 3D Extension

We are interested in this section in finding a minimal curve in a 3D image. One application that can motivate this problem is detailed in section 3.1. It can also have many other applications. Our approach is to extend the minimal path method of previous section to finding a path $C(s)$ in a 3D image minimizing the energy:

$$\int_{\Omega} \tilde{P}(C(s))ds \tag{2.1}$$

where $\Omega = [0, L]$, L being the length of the curve. An important advantage of level-set methods is to naturally extend to 3D. We first extend the *Fast Marching* method to 3D to compute the minimal action U. We then introduce different improvements for finding the path of minimal action between two points in 3D. In the examples that illustrate the approach, we see various ways of defining the potential P.

Similarly to previous section, the minimal action U is defined as

$$U(p) = \inf_{\mathcal{A}_{p_0,p}} \left\{ \int_{\Omega} \tilde{P}(C(s))ds \right\} \tag{2.2}$$

where $\mathcal{A}_{p_0,p}$ is now the set of all 3D paths between p_0 and p. Given a start point p_0, in order to compute U we start from an initial infinitesimal front around p_0. The 2D scheme equation (1.7) is extended to 3D, leading to the scheme

$$\begin{array}{rcl} (\max\{u - U_{i-1,j,k}, u - U_{i+1,j,k}, 0\})^2 & + & \\ (\max\{u - U_{i,j-1,k}, u - U_{i,j+1,k}, 0\})^2 & + & \\ (\max\{u - U_{i,j,k-1}, u - U_{i,j,k+1}, 0\})^2 & = & \tilde{P}^2_{i,j,k} \end{array} \tag{2.3}$$

giving the correct viscosity-solution u for $U_{i,j,k}$. The algorithm which gives the order of selection of the points in the image is detailed in table 2.1. It should be noted that a generalization of this algorithm was recently introduced in [92].

Considering the neighbors of grid point (i, j, k) in 6-connexity, we study the solution of the equation (2.3). We note $\{A_1, A_2\}$, $\{B_1, B_2\}$ and $\{C_1, C_2\}$ the three couples of opposite neighbors such that we get the ordering $U_{A_1} \leq U_{A_2}$, $U_{B_1} \leq U_{B_2}$, $U_{C_1} \leq U_{C_2}$, and $U_{A_1} \leq U_{B_1} \leq U_{C_1}$. To solve the equation, three different cases are to be examined sequentially in table 2.2. We thus extend the *Fast Marching* method, introduced in by *Adalsteinsson and Sethian* [2], and used by *Cohen and Kimmel* [34] to our 3D problem.

2.2 Several minimal path extraction techniques

In this section, different minimal path extraction procedures are detailed. We present new back-propagation techniques for speeding up extraction, a one end-point path extraction method to reduce the need for interaction, and in next section, a centering path extraction method adapted to the problem of tubular structures in images. The methods presented in this section are valid in 2D as well as in 3D and this is an

Algorithm for 3D Fast Marching

- Definition:
  - *Alive* is the set of all grid points at which the action value has been reached and will not be changed;
  - *Trial* is the set of next grid points (6-connexity neighbors) to be examined and for which an estimate of U has been computed using equation 2.3;
  - *Far* is the set of all other grid points, for which there is not yet an estimate for U;
- Initialization:
  - *Alive* set is confined to the starting point p_0, with $U(p_0) = 0$;
  - *Trial* is confined to the six neighbors p of p_0 with initial value $U(p) = \tilde{P}(p)$;
  - *Far* is the set of all other grid points p with $U(p) = \infty$;
- Loop:
  - Let $(i_{min}, j_{min}, k_{min})$ be the *Trial* point with the smallest action U;
  - Move it from the *Trial* to the *Alive* set (i.e. $U_{i_{min}, j_{min}, k_{min}}$ is frozen);
  - For each neighbor (i, j, k) (6-connexity in 3D) of $(i_{min}, j_{min}, k_{min})$:
    * If (i, j, k) is *Far*, add it to the *Trial* set and compute U using table 2.2;
    * If (i, j, k) is *Trial*, recompute the action $U_{i,j,k}$, and update it.

Table 2.1. *Fast Marching* algorithm

important contribution that can be useful for image analysis in general, for example in radar applications [8], in road detection [117], or in finding shortest paths on surfaces [86].

Examples in 2D are used to make the following ideas easier to understand. We also illustrate the ideas of this section on two synthetic examples of 3D front propagation in figures 2.1 and 2.2. Examples of minimal paths in 3D real images are presented in chapter 3.

The minimal action map U computed according to the discretization scheme of equation (2.2) is similar to convex, in the sense that its only local minimum is the global minimum found at the front propagation start point p_0 where $U(p_0) = 0$. The gradient of U is orthogonal to the propagating fronts since these are its level sets. Therefore, the minimal action path between any point p and the start point p_0 is found by sliding back the map U until it converges to p_0. It can be done with a simple steepest gradient descent, with a predefined descent step, on the minimal action map U, choosing

$$p_{n+1} = p_n - \text{step} \times \nabla U(p_n). \tag{2.6}$$

More precise gradient descent methods like Runge-Kutta midpoint algorithm or Heun's

Algorithm for 3D Up-Wind Scheme

1. Considering that we have $u \geq U_{C_1} \geq U_{B_1} \geq U_{A_1}$, the equation derived is
$$(u - U_{A_1})^2 + (u - U_{B_1})^2 + (u - U_{C_1})^2 = \tilde{P}^2 \tag{2.4}$$
Computing the discriminant Δ_1 of equation (2.4) we have two possibilities
    - If $\Delta_1 \geq 0$, u should be the largest solution of equation (2.4);
        - If the hypothesis $u > U_{C_1}$ is wrong, go to 2;
        - If this value is larger than U_{C_1}, go to 4;
    - If $\Delta_1 < 0$, at least one of the neighbors A_1, B_1 or C_1, has an action too large to influence the solution. It means that the hypothesis $u > U_{C_1}$ is false. Go to 2;
2. Considering that we have $u \geq U_{B_1} \geq U_{A_1}$ and $u < U_{C_1}$, the new equation derived is
$$(u - U_{A_1})^2 + (u - U_{B_1})^2 = P^2 \tag{2.5}$$
Computing the discriminant Δ_2 of equation (2.5) we have two possibilities
    - If $\Delta_2 \geq 0$, u should be the largest solution of equation (2.5);
        - If the hypothesis $u > U_{B_1}$ is wrong, go to 3;
        - If this value is larger than U_{B_1}, go to 4;
    - If $\Delta_2 < 0$, B_1 has an action too large to influence the solution. It means that $u > U_{B_1}$ is false. Go to 3;
3. Considering that we have $u < U_{B_1}$ and $u \geq U_{A_1}$, we finally have $u = U_{A_1} + P$. Go to 4;
4. Return u.

Table 2.2. Solving locally the upwind scheme

method can be used for this path extraction. A simpler descent can be choosing $p_{n+1} = \min_{\{\text{neighbors of } p_n\}} U(p)$, but it gives an approximated path in the L_1 metric. Such a descent has no more the property of being consistent. As an example, see in figure 2.1 the computed minimal action map for a 3D Homogeneous medium defined by $P(i, j, k) = 1 \; \forall (i, j, k)$.

Figure 2.2 shows a front propagation an a synthetic binary example, based on a spiral. We extract a path that goes from the interior of the spiral, and finds its way out of it to the second end point outside the object.

2.2.1 Partial Front Propagation

An important issue concerning the back-propagation technique is to constrain the computations to the necessary set of pixels for one path construction. Finding several paths inside an image from the same seed point is an interesting task, but in case we have two fixed extremities as input for the path construction, it is not necessary to propagate the front on all the image domain, thus saving computing time. Figure 2.3 compares the sets of pixels visited using a classical front propagation, and a partial

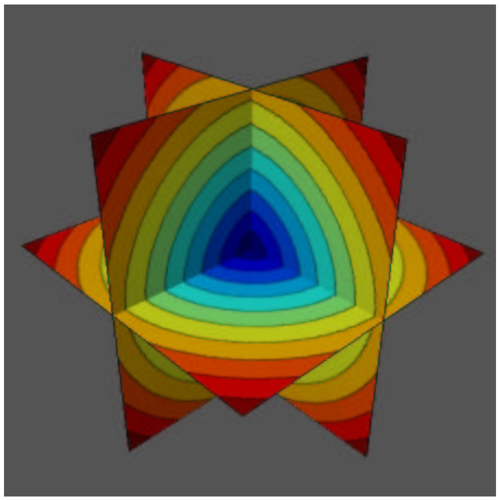

Figure 2.1. 3D visualization of an action map: these are the level sets of the action obtained by propagating a front in a homogeneous medium (constant Potential: $\forall (x,y,z) \in \mathbb{R}^3 \; P(x,y,z) = c > 0)$, represented with different colors. The domain is cubic, and the starting point for the wave equation is located at the center of the cube. The iso-action surfaces are concentric spheres with the starting point as center.

propagation on a Digital Subtracted angiography (DSA) image of the brain vessels. It highlight the fact that the set of points visited is smaller when propagation is only partial. We can see that there is no need to propagate further the points examined in figure 2.3-right, the path found being exactly the same as in figure 2.3-middle where front propagation is done on all the image domain. We used a potential $P(\mathbf{x}) = |\nabla G_\sigma * I(\mathbf{x})| + w$, where I is the original image (512^2 pixels, displayed in figure 2.3-left), G_σ a Gaussian filter of variance $\sigma = 2$, and $w = 1$ the weight of the model. In figure 2.3-right, the partial front propagation has visited less than half the image. This ratio depends mainly on the length of the path tracked.

2.2.2 Simultaneous Front Propagation

The idea is to propagate simultaneously a front from each end point p_0 and p_1. Let us consider the first grid point p where those front collide. Since during propagation the action can only grow, propagation can be stopped at this step. Adjoining the two paths, respectively between p_0 and p, and p_1 and p, gives an approximation of the exact minimal action path between p_0 and p_1. Since p is a grid point, the exact minimal path might not go through it, but in its neighborhood. Basically, there exists a real point p^*, which nearest neighbor on the Cartesian grid is p which belongs to the minimal path. Therefore, the approximation done is sub-pixel and there is no need to propagate further. This *colliding fronts* method is described in table 2.3.

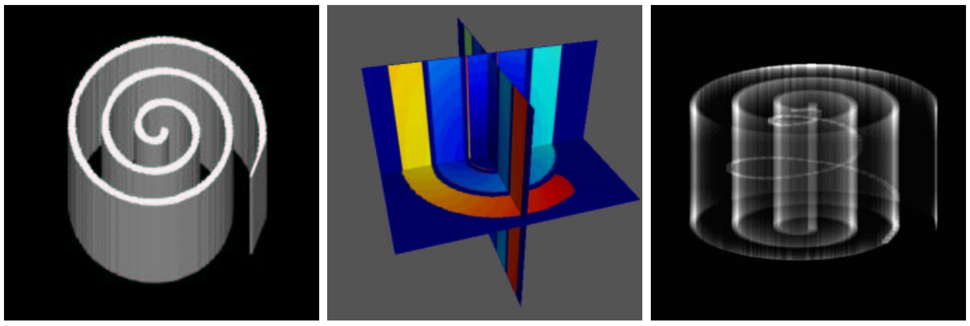

Figure 2.2. Front propagation in a synthetic 3D example: The left image is a volume rendering of the synthetic dataset, a spiral image, with a very high value ($P(x, y, z) = c_1 = 10^4$) in the spiral walls and a very low value in the background ($P(x, y, z) = c_2 = 1$). We extract the minimal path between a starting point located inside the spiral, and another one outside the spiral. The middle image represents the level sets of the action, mapped on three orthogonal planes, using the same color-map than in figure 2.1. The right image represents a transparent view of the object and the extracted path obtained.

Algorithm

- Compute the minimal action maps U_0 and U_1 to respectively p_0 and p_1 until they have an *Alive* point p_2 in common;
- Compute the minimal path between p_0 and p_2 by back-propagation on U_0 from p_2;
- Compute the minimal path between p_1 and p by back-propagation on U_1 from p_2;
- Join the two paths found.

Table 2.3. Minimal Path as intersection of two action maps

It has two interesting benefits for front propagation:

- It allows a parallel implementation of the algorithm, dedicating a processor to each propagation;
- It decreases the number of pixels examined during a partial propagation. With a potential defined by $P = 1$, the action map is the Euclidean distance.
  - In 2D (figure 2.5), this number is divided by $\frac{(2R)^2}{2\times R^2} = 2$;
  - In 3D (figure 2.1), this number is divided by $\frac{(2R)^3}{2\times R^3} = 4$.

Figure 2.4 compares the sets of pixels visited using a partial front propagation, as explained in section 2.2.1, and a simultaneous propagation on a Digital Subtracted angiography (DSA) image of the brain vessels. It highlight the fact that the set of

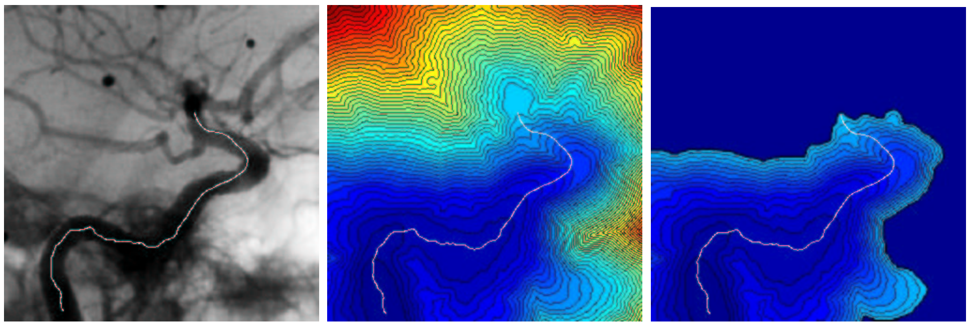

Figure 2.3. Comparing classical and partial propagation: The left image is the data set, used as potential to extract a path which stays inside a brain vessel of a DSA image; the extremities of the path are located manually. The center image is the action map obtained by propagating on the whole image domain. The right image shows the action map resulting from a partial computation. The two paths extracted are the same, due the minimality principle.

points visited is even smaller when propagation is done from both the extremities of the path.

The potential used is $P(\mathbf{x}) = |I(\mathbf{x}) - C| + w$, where I is the original image (256×256 pixels, displayed in figure 2.4-(a)), C a constant term (mean value of the start and end points gray levels), and $w = 10$ the weight of the model. In figure 2.4-(b), the partial front propagation has visited up to 60% of the image. With a colliding fronts method, only 30% of the image is visited (see figure 2.4-(c)), and the difference between both paths found is sub-pixel (see figure 2.4-(d) where the paths superimposed on the data do not differ).

The diagram in figure 2.5 represents the theoretical difference of domains visited by the algorithm, for partial and simultaneous propagations.

2.2.3 Euclidean Path Length Computation

We have shown the ability of the front propagation techniques to compute the minimal path between two fixed points. In some cases, only one point should be necessary, or the needed user interaction for setting a second point is too tedious in a 3D image. Here we derive a method that builds a path given only one end point and a maximum path length.

As we explain below, we can compute simultaneously at each point the energy U of the minimal path and its length. We choose as end point the first point for which the length of the minimal path has reached a given value. Since the front propagates faster along lower values of Potential, interesting paths are longer for a given value of U.

The technique is similar to that of section 2.2.1, but the new condition will be to stop propagation when the first path corresponding to a chosen Euclidean distance is extracted. Since the front propagates in a tubular structure, all the points for which

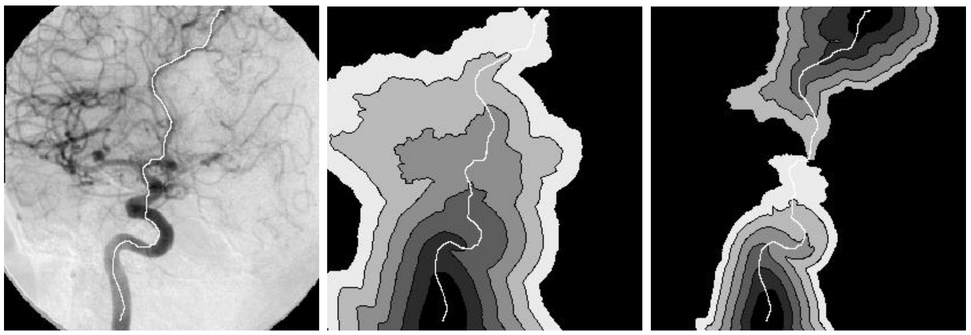

Figure 2.4. Comparing partial and simultaneous propagation: The left image is the data set, used as potential to extract a path in a vessel tree in a DSA image; the extremities of the path are located manually. The center image is the action map obtained by partial front propagating as explained in section 2.2.1. The right image shows the action map resulting from a simultaneous propagation from both extremities of the path. The second path extracted is a sub-pixel approximation of the first one, as detailed in the section.

the path length criterion is reached earlier in the process are located in the same area, far from the start point. Therefore the first point for which the length is reached is located in this area and is a valuable choice as endpoint.

Figure 2.6 represents the propagation of a front according to the potential shown in figure 2.6-left, with the *on-the-fly* computation of the approximate Euclidean length of the paths at each pixel crossed by the front. The propagation is done on the whole image domain, and one can observe that the resulting map, in figure 2.6-right is non-smoothed, and very difficult to analyze.

Figure 2.7 represents the same computation of the Euclidean path length than in figure 2.6-left, but limited to the necessary set of pixels visited in order to extract the minimal path super-imposed on the three images (the method is detailed in section 2.2.1). One can observe that the resulting map, in figure 2.6-right is non-smoothed, but we can clearly visualize the level-sets of the Euclidean path length computed at the same time.

2.3 Path centering in linear objects

The path is the set of locations that minimize the integral of the potential in equation (1.3). If the potential is constant in some areas, it will lead to the shortest Euclidean path. The same thing happens when the potential does not vary much inside a tubular shape. The minimal path extracted is often tangential to the edges, and would not be tuned for a problem which may require a centered path. Figure 2.8 describes the practical problem we are facing using the classical wave equation model of [34], in tube shaped structures where the potential is approximately homogeneous inside the object.

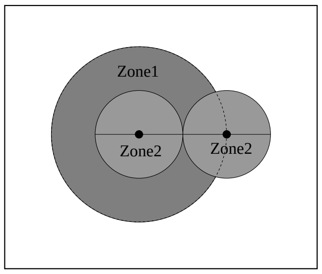

Figure 2.5. Comparing the theoretical domains for extracting a minimal path between two points in a homogeneous medium: The area labeled *zone 1* corresponds to the needed set of pixels visited with partial propagation. The area labeled *zone 2* corresponds to the needed set of pixels with simultaneous propagation.

The general framework for obtaining a centered path is the following

- Segmentation : the first goal is to obtain the edges of the tubular region;
- Centered path : once we have this segmented region, we want to find a path that is as much centered as possible in it. In order to attract the minimal path to the center of the region, we use a distance map from the segmented edges.

In the following we are going to present our method, introduced in [42], detailing each step and making comparisons with other existing techniques.

2.3.1 Segmentation step

In order to find the tubular structure, several approaches can be used. We can use a balloon model [29] with a classical snake approach that inflates inside the object, starting with the given end point. Or we can segment the object using its correspondent level-sets implementation, as in [113] and like the bubbles in [172]. In fact, this kind of region growing method can also be implemented using the *Fast Marching* algorithm. This fast approximation has already been used for segmentation in [111]. This allows us to include the segmentation step in the same framework as our minimal path finding: having searched for the minimal action path between two given points, using a partial front propagation (see section 2.2.1), the algorithm provides different sets of points:

- the points whose action is set and labeled *Alive*;
- the points not examined during the propagation and labeled *Far*;
- the points at the interface between *Alive* and *Far* points, whose actions are not set, and labeled *Trial*.

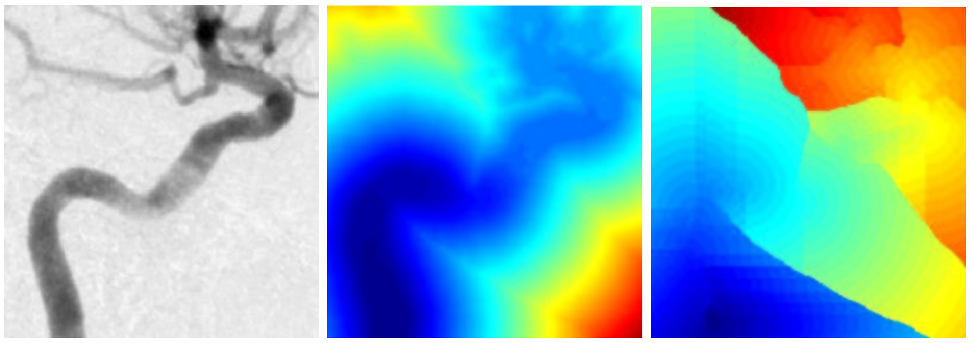

Figure 2.6. Computing the approximate Euclidean path length while propagating on the whole image: using the left image as potential, the front is propagated on the whole image domain. Middle image and right images represent respectively the action map starting from the bottom of the vessel, and the Euclidean path length computed at the same time.

This last category, the border of the visited points, is a contour in 2D and a surface in 3D which defines a connected set of pixels or voxels. If the potential is a lot higher along edges than it is inside the shape, the edges will act as an obstacle to the propagation of the front. Therefore, the front propagation can be used as a segmentation procedure, recovering the object shapes. In this case the *Trial* points define a surface which can be described as a rough segmentation. Once the front has reached the endpoint, we use the front itself to define the edges.

2.3.2 Centering the path

Having obtained this interface of *Trial* points, we now want the information of distance to the edges. This information can be either used for a skeletonization, computing the medial-axis transform, or used as a new snake energy, that constrains the path in the center of the tubular shape.

In order to compute this distance, we can use a second front propagation procedure. The edges ares stored in the min-heap data-structure (see [163] for details), and this is a very fast re-initialization process to compute this distance. The potential and the initial action for this second front propagation are defined as follows:

$$\begin{aligned} P(i,j) &= 1 && \forall (i,j) \text{ inside the shape} \\ P(i,j) &= \infty && \forall (i,j) \text{ outside the shape} \\ U(i,j) &= 0 && \forall (i,j) \in \{\textit{Trial}\} \text{ points of section 2.3.1} \\ U(i,j) &= \infty && \text{elsewhere} \end{aligned}$$

Starting the front propagation from all the points stored in the min-heap data-structure, we compute the distance map, said $\mathcal{E}$, very quickly, visiting only the pixels inside the tubular object.

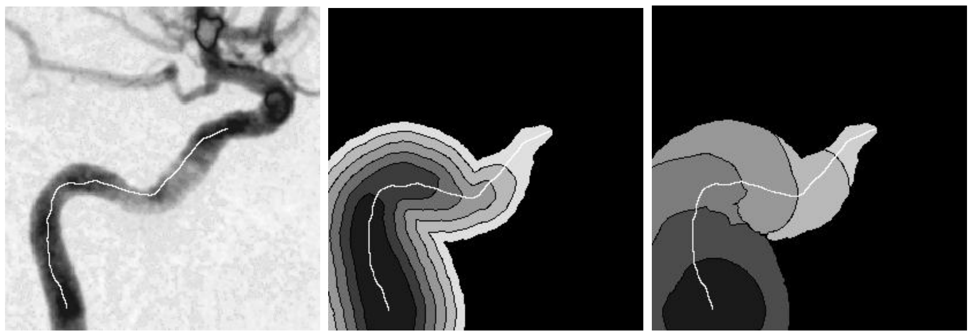

Figure 2.7. Computing the approximate Euclidean path length while propagating partially on the image domain: Left image is the potential. Middle image and right images represent respectively the action map starting from the bottom of the vessel, and the Euclidean path length computed at the same time.

Our distance map $\mathcal{E}$ is used to create a second potential P_1. Choosing a value d to be the minimum acceptable distance to the walls, we propose the following potential:

$$P_1(\mathbf{x}) = \max\left(d - \mathcal{E}(\mathbf{x}); 0\right)^{\gamma} \tag{2.7}$$

This distance d is illustrated on figure 2.9. We use this potential (2.7) for a new front propagation approach: P_1 weights the points in order to propagate faster a new front in the center of the desired regions. This final propagation produces a path centered inside the tubular structure in a very fast process.

2.3.3 Description of the method

The complete method is described in figure 2.10.

1. Segmentation: the first step is to compute the weighted distance map given the start and end points. It is obtained by front propagation from the start to the end point. Notice also that the end point can be determined automatically by a length criterion as in section 2.2.3;

2. Segmentation: the second step is to consider the set of points which have same minimal action as the endpoint. For this, we store the front position (set of trial points) at the end of the first step.

3. Centering Potential : the third step is to compute the distance map $\mathcal{E}$ to the boundary front inside the tubular region. For this we propagate inward the front with a uniform potential $P = 1$. This gives the higher values towards the center of the object.

4. Centered path : the fourth step is to find the minimal path between start and end points relatively to the distance potential P_1 defined in (2.7) computed from the previous step. This is obtained by applying again the minimal path

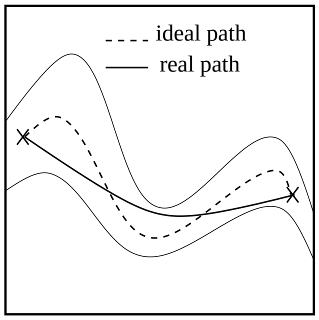

Figure 2.8. Path extraction in a tube-like object: the path obtained using the classical wave equation model is minimal according to the weighted metric, which means that it is the shortest in the tube considered; the ideal path would stay in the center of the object, as shown in the diagram.

technique. The front is now pushed to propagate faster in the center of the object.

5. Centered path : the final step is to make back-propagation from the end point using the last minimal action map.

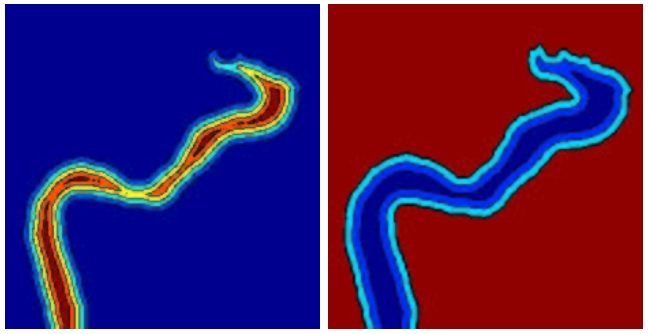

Figure 2.9. Thresholding the distance map: The left image shows the level sets of the distance to the object borders; The right image is the potential obtained by applying a threshold to the distance in order to propagate faster in a region of the tube, at a minimum distance to the borders, given as parameter.

Figure 2.10 explains the different steps of the path centering process.

An interesting improvement is that the value of the weight w can be automatically set to a very low value:

- During the first propagation the regularity of the path is not important, and w can be very small;

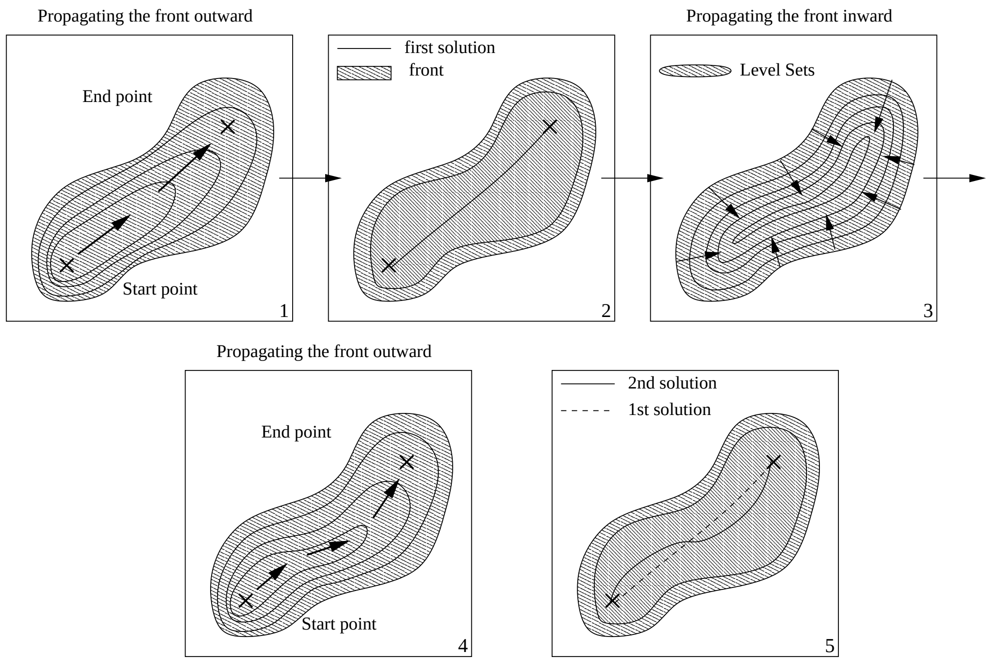

Figure 2.10. The complete path centering process: step 1 is propagation in the medium between the two extremities of the path; step 2 is to consider and store the envelope of the pixels visited during step 1; step3 is to propagate backward into the object, computing the distance to its borders; step 4 is to use the distance as a new penalty to propagate; step 5 is to backtrack the final centered path.

- During the second propagation, $P' = P + w = 1$
- During the final propagation the potential based on the distance to the object walls is synthetic and leads to smooth paths even if $w << 1$.

Figure 2.11 compares the result of the classical path extraction, and the centering process detailed below, on a potential defined by a DSA image of the brain vessels. The two paths are represented super-imposed on the data in figure 2.11-left, in order to highlight the result of the modification of the penalty according to the distance to the object.

In figure 2.11-middle is shown the result obtained using a potential based on the image, where the shortest path is tangential to edges. But the front propagates only along the vessel direction, and is rapidly stopped transversally, allowing to compute the distance to the walls. Defining a new potential according to equation (2.7) based on this distance map, the second front propagates faster in the center of the vessel, at the distance d chosen. Due to the shape of the iso-action lines of the centered minimal action shown in figure 2.11-right, the path avoids the edges and remains in the center of the vessel.

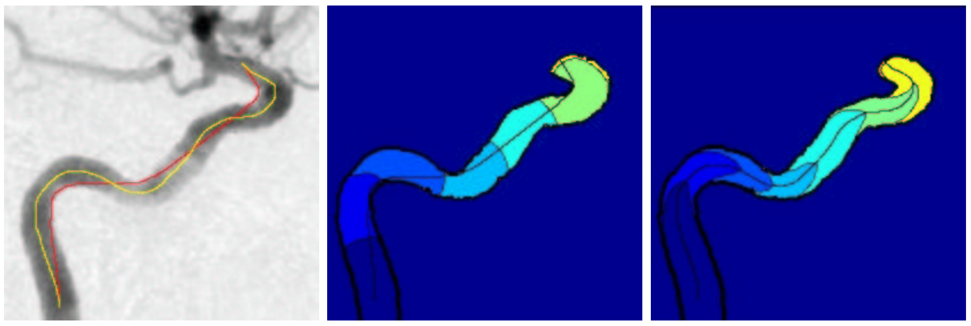

Figure 2.11. Path centering test on a 2D DSA image: Applying the process detailed in figure 2.10, the two paths extracted are super-imposed on the data on the left image; the middle image represents the level-sets of the action map with a classical use of the Eikonal equation; the right image shows the corresponding result with the centering strategy.

2.3.4 Comparison with other work

Another method to obtain a centered path would be to make a classical snake minimization on the centering potential P_1, starting from the path obtained previously, like it is done in the thesis [36]. But too much smoothing may lead to a wrong path. For example, in the case of thin tubular structures, smoothing the path may lead it outside the tubular structure. Also, the unpublished work presented in [36] details an algorithm which is applied to a tubular object which is already manually segmented by the user, whereas our method comprises both steps of segmentation and centering.

Another category of very similar centered line extraction technique is skeletonization, and particularly the definition of the medial axis function of [15] which treats all boundary pixels as point sources of a wave front. Considering that the *Fast Marching* computes the Euclidean distance to an arbitrary set of points using a potential $P = 1$, it can also be used for skeletonization.

However, the purpose of our application is to have a smooth line which always stays inside the tubular object and which is far from the edges.

If one wishes to achieve this task with a skeletonization, like in [192], he will need and rely on the results of post-processing techniques in order to obtain a unique and smooth path inside this segmented object. Smoothing and removing undesirable small parts of the skeleton can be done using techniques shown in [173]. The main advantage of our approach is that it gives only one smooth and centered path in a unique and fast process. Therefore, it cannot be replaced by a simple medial-axis transform.

In [136], the authors extract first the surface of the colon, then compute a minimal path on this surface and move this initial path to the center of the object by applying a thinning algorithm to the object segmented and projecting the path on the resulting surface. The algorithm developed by [90] can be applied to their methods since it computes the minimal path on a surface defined by a manifold. Although it seems to produce a smooth centered line, the thinning algorithm is computationally inefficient, compared to the speed of our algorithm that needs less than a minute on a classical inexpensive computer (300MHz CPU).

In the different techniques quoted, the main difference with our method lies in the fact that the object is manually segmented by the user. Our method comprises steps of segmentation and path extraction, and achieves them in a very fast way. More than a robust and fast method, we have developed a tool that is used for segmentation, minimal path tracking, and even potential definition The main advantage of our approach is that it comprises all those steps and gives only one smooth and centered path in a unique and fast process.

2.4 Introducing the angle as a dimension

2.4.1 Principles

In [92, 91] the authors consider the problem of robot navigation with constraints and rotation, introducing a third degree of freedom in two-dimensional applications. Considering now an object with a given length and width, the problem is now to extract a trajectory between two positions that are in configuration space the position of the center of the object, plus an angle θ between 0 and 2π at the beginning and at the end of the trajectory.

The authors consider two cases, both on constant potentials:

- In the absence of obstacles, the *Fast-Marching* can be applied in a straightforward manner, by discretizing the configuration space into a 3D grid, namely griding both $\mathbb{R}^2$ and $[0; 2\pi]$ with periodic boundaries in θ, and solving

$$\left[u_x^2 + u_y^2 + u_\theta^2\right]^{\frac{1}{2}} = 1 \qquad (2.8)$$

- In the presence of obstacles, by altering the shapes of the obstacles for every discretized angle θ_i, rather than maneuvering the robot. They use morphological operations, like dilatation to alter the obstacles shapes, with a structuring element corresponding to the robot at a given angle. A fast implementation of these morphological operations can be found in [64].

Therefore the above path planning problem solves the Eikonal equation

$$|\nabla T| = \frac{1}{V(x, y, \theta)} \qquad (2.9)$$

where F is binary: 1 in reachable regions and 0 inside the obstacles. As shown in [163], there is no reason to limit ourselves to binary speed values. We may use the same algorithm for continuously varying speed functions.

In [88], they consider the problem of obstacle avoidance navigating under a potential function which penalizes the free work space [100].

We worked upon the use of the Eikonal equation, including a dimension related to the angle of an object, in order to compute trajectories of oriented objects in domains with a weighted metric, without obstacles.

The problem has been schematized to the following:

1. We have used very simple objects, like rectangles and triangles, in two dimensional media;
2. For those objects, our strategy is to discretize them in a limited number of positions, which means that for a triangle, we only consider the value of the potential at its vertices.
3. In order to make the objects move according to the weighted metric, staying in the desired regions, at each position (x, y, θ), we take as potential for an object, the maximum of the potential over all positions considered (i.e. vertices for the triangle case).

We want to simulate the trajectory of an object, in a two dimensional medium, with constraints on the direction of the object: there is now a cost for changing its orientation. This result finds its application in the regularization of the point of view of the object in the media, simulating for example the direction of a virtual camera.

2.4.2 Algorithmic tricks

The algorithm used is very similar to that of section 2.1, which means we are working on a three dimensional problem. The following definitions are necessary:

1. The speed of the front is a function of the position, but not the orientation: $V(x, y, \theta) = V(x, y)\ \forall (x, y, \theta) \in \mathbb{R}^2 \times [0; 2\pi]$;
2. The action computed according to equation (2.9) is defined on $\mathbb{R}^2 \times [0; 2\pi]$, with periodic boundaries in θ.
3. the griding of the interval $[0; 2\pi]$ used in the tests was to consider 10 discretized angle; it can be easily implemented using an array.

2.4.3 Results

Figure 2.12 represents samples of the trajectory of a triangle, using a DSA image as weighted metric for propagating. Obviously, the object is doing several U-turns along the trajectory and is not suitable for the applications we want to address.

We overcome the drawback of the very simple object used in figure 2.12-left by simply adding a branch to our triangle. This branch defines a new position for estimating the potential, and will constrain our object to look in the direction of the trajectory, in linear structures, like vessels. Results of this new strategy are shown in figure 2.12-right.

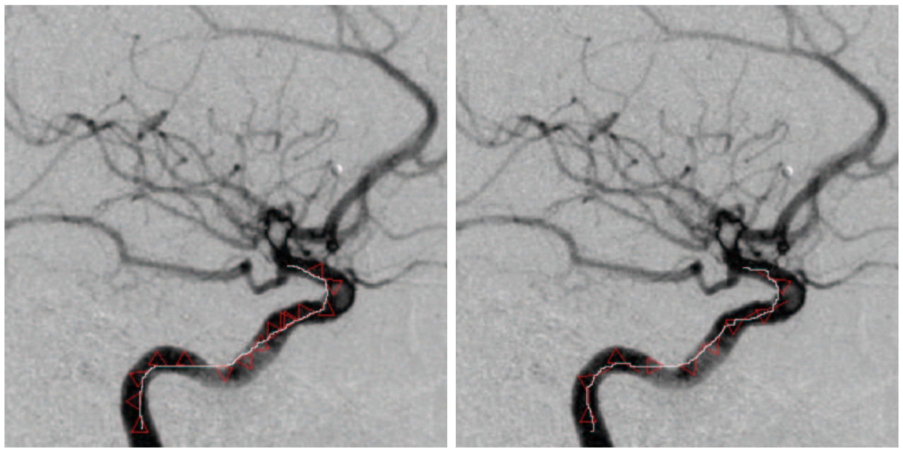

Figure 2.12. Movement of a two different objects in the medium defined by the image on the background: Left image: the object is a triangle; the path represents the trajectory of one of the vertices of the triangle. The constraint on the angle of the object is not sufficient, and the object is doing several U-turns during propagation; right image: the object is now a triangle with a branch connected at one of its vertices; the constraint on the angle of the object is now sufficient, and the object keeps looking in the direction of the trajectory.

2.4.4 Perspectives

Linear objects with self-intersections: in a 2D X-ray image, a linear three dimensional structure projection can self-intersects, like a catheter in a heart image (for example see figure 2.13). The minimal path using only the 2D spatial configuration will not extract the loop which is created. Our method could overcome this drawback.

2.5 Introducing recursivity in the Eikonal equation

The *Fast-Marching* algorithm fails if the penalty is noisy, or if the objects to detect are long thin curves, like the guide-wire shown in figure 2.14. If the offset term w and the penalty $\mathcal{P}$ in *Eikonal equation* are not tuned efficiently, a portion of the shortest path extracted can be a short cut to the starting point, leading to wrong results, like in figure 2.14-left. One way to overcome this drawback is to introduce a recursivity term in the computation in *Eikonal equation* : having $\alpha \in]0;1]$, we now compute

$$\begin{aligned}(\max\{u-\alpha.\mathcal{U}_{i-1,j,k}, u-\alpha.\mathcal{U}_{i+1,j,k}, 0\})^2 &\quad + \\ (\max\{u-\alpha.\mathcal{U}_{i,j-1,k}, u-\alpha.\mathcal{U}_{i,j+1,k}, 0\})^2 &\quad + \\ (\max\{u-\alpha.\mathcal{U}_{i,j,k-1}, u-\alpha.\mathcal{U}_{i,j,k+1}, 0\})^2 &\quad = \quad \tilde{\mathcal{P}}^2_{i,j,k}\end{aligned} \tag{2.10}$$

giving the value u for $\mathcal{U}_{i,j,k}$. This recursive term, usually set to .9, reduces the values of the action. This enables the front to propagate further in the direction of the thin

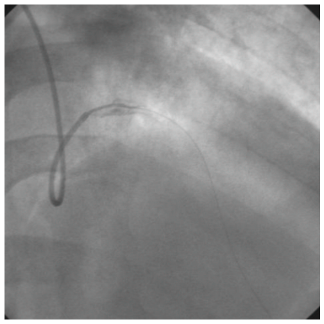

Figure 2.13. Catheter in X-Ray image of the heart: The catheter is the linear structures in dark that self-intersects.

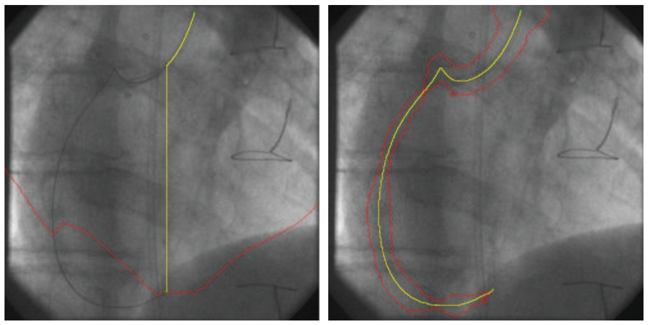

Figure 2.14. Guide-wire extraction with recursive *Fast-Marching* : The left image is the minimal path extraction with classical *Fast-Marching* ; right image is the same result with recursive *Fast-Marching* .

curves, without propagating in all directions. In figure 2.14, the border of the set of visited points is drawn in red: on figure 2.14-left the algorithm has visited more than half the image domain, leading to a wrong path which goes straight down to the end point; on figure 2.14-right, the corresponding domain surrounds the guide-wire, and leads to a path that stay in the vicinity of the object.

Unfortunately, the equation(2.10) of course no more gives the solution to any *Eikonal equation* computed on a penalty domain $\mathcal{P}$, and the new action map U computed is not convex at all. The minimal path that links the extremities is here defined by a L_1 descent on the map which stores the *Fast-Marching* iterations, and not with the gradient descent on the action map. The minimality principle is lost, whereas this algorithmic trick improves extraction. A patent has been filled on this subject (see [54]).